HOT Z-AROS

USER'S NOTES

SINWARE
Box 8032
Santa Fe, NM 87504

THIS IS HOT Z-AROS

The 2Ø68 is a machine that is well out of its price class in potential for development, well below it in realization. What's 253 times 64K, expressed in the current price of dynamic RAM chips? (Ridiculous, that's what.) But HOT Z attempts to play along.

These notes are for HOT Z users, who are liable to be sick of notes, or no longer to have time for them. Read the first section and save the second for reference. You should especially read about the new Transfer command later in this section.

For vocabulary, I recommend the Timex technical manual, if you can get it, though many complain. I have tried to develop and make function just what was put into this machine by its designers. This version of HOT Z is intended to distinguish between the Spectrum and the 2Ø68. I believe that many owners have retreated to the Spectrum for its wealth of software. But the 2Ø68 has something to add, so hang on to your ROMs. The 2Ø68 is not a Z8Ø Atari. It is a 3 x 64K Z8Ø memory manager that could be a CP/M machine with all the memory you would want for this processor.

There is resident software for handling 253 other memory banks, but more hardware is required to manage more than the three 64K banks. That was the intention in the Bus Expansion Unit (BEU), which, according to rumor, was breadboarded or even demoed in a discrete chip version, but never produced. Now if someone will please wire up a disk controller that runs off the bus...

The Timex description of available memory is as follows:

- The 16K of ROM and 48K of RAM that you work with normally are called the 64K Home bank, numbered FF
- The 8K called the ROM Extension or EXROM is the bottom chunk of another 64K bank that can be selected on the expansion bus via EXROM NOT, numbered FE
- The cartridge bank is 64K and is available either on the Dock connector or the bus via ROSCS NOT, numbered Bank ØØ
- Other banks or "devices" are numbered Ø1 to FD and require the BEU

A high proportion of the 2068 is intended to be ROM, but none of it appears to be necessesarily so. It will require far less ingenuity to expand the 2068 to 192K than it did to bring the ZX81 up to an active 64K.

In addition to the Banks, memory is further divided into chunks, a division referred to as "horizontal." A chunk is 8K within a 64K Bank, so that there are 8 chunks in every 64K bank. The eight chunks are numbered Ø to 7, and in hex the addresses run from ØØØØ to 2ØØØ to 4ØØØ to 6ØØØ to 8ØØØ to AØØØ to CØØØ to EØØØ and finish with FFFF in chunk 7.

The 2068 code allows you to "enable" any eight chunks within the available memory. Only 64K can be active at once. The 2068 describes the active chunks for any bank with a byte called a chunk spec, which consists of eight binary places in which a 1 means "inactive" and a Ø means active or enabled. Hex CF in binary is 11ØØ1111, which means the 4th and 5th chunks, starting with Ø and from the right, are the ones to be enabled. (This is the normal enablement of the Dock bank when HOT Z is running.)

The bank number takes another byte, which usually goes into B, while the chunk spec or "horizontal select" byte goes into C. If you then CALL ENAB, you enable the selected chunks of the given bank within your active 8 chunks.

DISB -- A New HOT Z Variable

The new HZ system variable, DISB (disassembly bank), must be manipulated directly by the user to control what you see with the disassembly and what memory you change with HZ commands. DISB is a two-byte variable that is actually a bank-chunk spec; the high byte is the bank (FE = EXROM, FF = HOME, ØØ = DOCK) and the low byte is the active-low chunk-enable byte (ØØ enables all chunks, FE enables chunk Ø, 7F enables chunk 7, etc.) The default on startup is FFØØ, which is all chunks of the HOME bank and leaves HOT Z itself invisible. (To read HOT Z, you would change DISB to ØØCF, turning the disassembler on for chunks 4 and 5 of the DOCK bank, and leaving it on HOME for the other chunks. To read the EXROM, you would enter FEFE to DISB.)

All of the values mentioned can be written in directly, but there are a few combinations that hang the machine. All zeroes, for example, mean enable the dock everywhere, which locks out the stack, as does any combination of bank and chunk spec that turns off a chunk 7 that is not RAM with the stack in it. Use the legal values listed, and experiment with others when you have nothing cassette-loaded, and reset is just a flick of the switch away.

Valid combinations of bytes for DISB will depend on what you have connected to the 2Ø68. If you can hook up a chunk Ø in some bank, then you should have an interrupt fielder at ØØ38 as a minimum before you enable such a bank without a DI. You can copy out the code from ØØ38 to ØØ48 in the EXROM if you need a fielder. Chunk 2 contains the system variables and the HOT Z RAM-res code, and you will have to come up with a smart, but not immensely smart, routine to make use of that chunk. Finally, chunk 7, from EØØØ to FFFF, contains the stack, and that must be moved to an active RAM chunk before you can switch out the Home RAM chunk 7.

Awkward values for DISB can generally be avoided by replacing them backwards (high byte first) or by using the Transfer command to move two bytes into DISB together.

Memory Occupation

The HOT Z program occupies 16K from 8ØØØ to BFFF. Included are a single Help screen (CSS-H or SQR in READ or EDIT) with the commands for all but the single-step, and a file of system variable NAMEs. Due to lack of memory space, not all of the commands that use the same key in both modes (Step, To, Then) are listed in the EDIT column.)

The channels area is at 7BØØ, and PROG and the various pointers start at 7B16. This leaves the second display file clear, although it is not currently in use. The location is much higher than you may be accustomed to, so if you have any usable addresses in mind from other systems or HZ's, check again. BASIC programs that have embedded machine code will probably require revision of the USR call addresses.

HOT Z-AROS has its variables and buffer area in RAM at 5F6Ø-5FFF. This could ultimately get in the way of the Syscon parameter table for memory banks and intelligent devices, but there is room for four or five, which should do for the near future. HOT Z uses a RAM-resident block of code, which is presently located between 5EØØ and the Syscon table at 5EEA. This could cause conflict with other devices or programs that use the same area. HOT Z continues to use much of the 5DØØ-5EØØ area for various buffers. Your workspace in RAM runs from 5Ø bytes above STKEND to the lower limit of the stack at about F7CØ. So long as you are working with a single display file, you may also use 6ØØØ to 7BØØ.

NAMEs

Since the NAME file is in EPROM, you must move it into RAM with the NSET command (INKEY$ in READ) before you try to add to it. The file is moved high, up next to the stack. Use the

RDN command in READ to find the start of the file. After you move it to RAM, you can put it anywhere there is space. The variable ALNA is included to assist switching file locations.

If you try to erase a NAME while the file is in EPROM, you will confuse the look-up and lose the use of the entire file until you reinitialize.

The labelling system has not been partitioned to be multi-bank. A NAME shows up at its address no matter what bank you are in. Since it is easy to switch between alternate files, I do not plan to refine this facility.

RAM-Res Corrections

This version corrects the Timex RAM resident code and relocates it to its high location, as if both display files were open. (The corrections differ from those in the tech manual and are better.) In order to get correct error trapping in the EXROM, it is necessary to set the system variable VIDMOD to a non-zero value. This does not itself have any effect on the display. The stack is also moved high and the variable MSTBOT is set to match. Since there is extensive use of the XFER bytes routine, the stack can often go to its full size, so don't attempt to preserve any information or code above F7CØ. (You should be able to go right up to that limit.)

If you write software that makes use of the bank-switching routines, you should get or make yourself a corrected EXROM to enable that software to run without HOT Z in the machine.

Upper/Lower Case

Since HZ does not recognize lower case for hex input nor the main part of a mnemonic, it can be inconvenient or even puzzling to be in that shift state on an RGB monitor with no bright cursor to indicate what is happening. Therefore, I have installed a few automatic turn-offs of the lower-case state: after entering a new NAME, after entering an assembly line, and on turning on Hexedit. The shift state does persist if you enter a lower case NAME to the top line cursor in READ mode.

NEW COMMANDS

Most of the new commands in HOT Z-AROS deal either with the fact that the program is in firmware or with bank switching.

First of all, in each of the three modes (READ, EDIT, STEP), the PI and the TAB keys can be hooked to your routines in RAM to turn them into HOT Z commands. All you do is write the address if your routine at the appropriate address. Those are as follows:

READ:	PI	5F90
	TAB	5F92
STEP:	PI	5F94
	TAB	5F96
EDIT:	PI	5F98
	TAB	5F9A

It is no longer possible to write an address to the command file, since the command file is in EPROM. You can also run routines, as always, by just putting the cursor at the start, crossing your fingers very tightly, and hitting CSS-R, but it is often more convenient to have them as commands. Unfortunately, space did not allow a routine that would handle bank switching for these hookups. As a result, the routine that you hook up must be in normally enabled RAM, which is to say RAM that is outside the range 8000-BFFF. You can enable and call into that area with CALL 5E05, CALL YOUR_ROUTINE, JP 5E00, but those 9 bytes must lie outside 8000-BF00. All other commands handle these details for you.

INKEY$ (NSET) from READ mode will read out the EPROM-resident NAME file and put it in RAM so that you can add to it. You are still responsible for keeping tapes of your own label files.

When you go to BASIC with SS-Q (<=) in READ, the memory as seen by BASIC at 8000-BFFF is then RAM. To get back to the cartridge, use RAND USR 24098.

The SCREEN$ command in READ will put you in 64-column mode but gives you no facilities for printing to the second display file. The HOT Z cursor also disappears. To get out, use the new POINT (CSS-SS-8) command, which restores 32 columns.

New Transfer Format

The transfer command is now thoroughly interbank. This means you have an extra parameter to specify, called S/D Banks (as in source/destination), but there is a default on this entry, so that if you hit ENTER you automatically set Home-bank-to-Home-bank and then go on to set the DEST. You are expected to

enter four digits, comprising two valid Bank specifiers. At the moment, the only valid choices are FE for Extension ROM, FF for Home Bank (BASIC ROM plus 48K RAM), and ØØ for Dock Bank (cartridge).

Since there is not much use trying to transfer to ROM or EPROM, the second specifier should always be FF until someone comes up with some banked expansion memory. Valid combinations for now are FFFF (Home/ Home), FEFF (Extension/ Home), and ØØFF (Dock/Home). FFFF is the default if you just press ENTER; the DEST prompt that follows is the usual destination address prompt. So for normal use just add an ENTER as the first key you press in the command sequence after CSS-T.

All transfers now go through the Timex RAM-res code, and that imposes one new restriction on moving memory contents. Since memory is enabled in 8K chunks, your transfer commands should never cross a chunk boundary (2ØØØH, 4ØØØH, 6ØØØH, etc.). If you want to transfer the contents of 9FFØ to AØ1Ø, for example, you should first move 9FFØ to 9FFF, then AØØØ to AØ1Ø. However, you can disregard this if the transfer is from RAM to RAM within the Home bank. (Business as usual.)

In previous versions of HOT Z, you could set the cursor and END in either order to block out the code to be moved, but the lower address was always the one whose contents went to DEST. Oftentimes, however, you may want to put the higher address to DEST and let the lower ones fall where they may. So now if you set the cursor to 7F1Ø and END to 7FØØ and then enter 7F9Ø to DEST, the transferred block will reside at 7F8Ø to 7F9Ø. After the command, the display is set to start with the lowest address of the transferred block.

A full list of commands follows for reference.

THE COMMAND SET

Keying is described as CSS- for the Caps/Symbl-Shift combination before another keystroke and SS- for Symbl Shift pressed simultaneously with another key. Keys are referred to by any of the three rubrics on the keytop. Mnemonic associations are generally with the letter on the key: for example, Assembly is Symbol-Shift/A, the STOP key. There is a brief help screen that you can call up from READ or EDIT modes with CSS-H (SQR).

READ Mode

Key	Description

QUIT TO BASIC

SS-Q — Quit HOT Z for BASIC. HOT Z and the entire Dock bank are switched out so that BASIC sees only Home bank

COPY

CSS-COPY — Copies the screen to the 2040 printer. Gives you headings and all. Consider using the LLIST command from an edit mode for no headings and variable length.

HEXEDIT

SS-E — Sets the cursor to the top line and switches to the hex-edit mode. This command also works from assembly-edit mode without resetting the cursor line.

ASSEMBLE

SS-A — Sets the cursor to the top line and switches to the assembly-edit mode. The same keystrokes will get you from hex-edit to assembly edit. This command works only when the disassembly display is on.

TOP NAME

CSS-T — Move the display to the 'top' of the NAME file and switch to the data display. Use this command as preparation for SAVing a NAME file. (Turn on the cursor, set END, and SAVE.) If the file is still in EPROM and DISB is set to its default, you will see the corresponding memory space in RAM, which may be empty or hold something else.

NAME SWITCH

CSS-SS-N (OVER) NAME file switch. If you are using only one file, the NAMEs are switched off or on. If you have two files in memory, the command will switch from one file to the other. Before switching, you must first write the start and end addresses of the new file at ALNA (lo-hi order). The end address is the first of two bytes of zeroes at the top end of the NAME file. To start a new file, set both addresses the same, pointing to two bytes of zeroes, then add names to the disassembly.

RESTART

CSS-R Restarts HOT Z. Resets the stack to clear clutter. Resets register values in the single step and sets the EPROM-resident NAME file active.

MAKE REM

CSS-REM Installs a 1 REM statement in BASIC at the value in the system variable prog (normally 7B16H). The REM will run to the value in END and will push other BASIC lines to higher memory.

BORDER-INK-PAPER

CSS-SS-BRIGHT BORDER color set. Follow with a color key.
-INK INK color set. Follow with a color key.
-PAPER PAPER color set. Follow with a color key.

STEP

SS-STEP Switch to single-stepper. The address in the NEXT and LAST slots will be last ones used there. Use this command to get back after an interruption. All old single-step register values are preserved.

DIS/DAT

SS-GOTO (THEN) The display switch from disassembly to data display or back again. The same command works with the hex-edit cursor on but not from assembly-edit.

SET END

SS-TO Enter a value to the END variable, as in EDIT mode, but the value is not displayed

DECIMAL ADDRESS

SS-OR Indicates decimal address to follow. Clears away the ADDR cursor and waits for your entry. If the decimal address is less than five digits long, hit ENTER after the last.

SCROLL

SS-<> Sets the screen to a continuous SCROLL. BREAK will stop it. A toy.

SP ON

SS-AT Toggles on or off a display of the machine stack-pointer address in the upper right screen corner. The default is Off, because it isn't pretty, but you should turn it on when you are test running your own routines. There is a small amount of shock absorption in the HOT Z stack, but if you should see it changing, then look very carefully at what you are doing to the stack with the routine you are testing. Restarting HOT Z will reset the stack.

FP IN-OUT

CSS-O (PEEK) Switch the on-off state of the floating-point disassembler. If turned off, then the SS-I command will have no effect. If on, then every EF (RST 28) will switch to the floating-point disassembly and every 38H will switch off the floating-point disassembly. If you have a stray EF on screen while you are in an edit mode, you may get a messed up display when you enter code. If so, exit (ENTER) from edit mode, use this command, and go back into the active mode without fear. Default state is OFF.

FP INTERPRETER SWITCH

CSS-I (CODE) Floating-point interpreter switch. This is a flag switch (NOT an on-off switch) which switches interpretation of a byte from Z80 language to floating-point language. This command is necessary for certain embedded sections of floating-point code that are not preceded by an RST 28 but are jumped to from some other portion of floating-point code. This command will not function if the PEEK switch has been set to off. If it doesn't work, hit PEEK and try again.

64 COLUMNS

CSS-SS-K (SCREEN$) Switches the display mode to 64 columns and will put noise on screen if you have anything in the second display file (6000-7B00). There is no provision for printing to the second display file, but it is there for you to work on. Intended as a development tool.

32 COLUMNS

CSS-SS-8 (POINT) Switches back to 32-column display. Intended as an escape from the 64-column mode.

NSET

CSS-N Moves the EPROM-resident NAME file into RAM, where it can be added to, changed, etc. The relocated file goes to high RAM. Use the Top NAME command to locate the beginning of the file.

HOOKUPS

CSS-M User hook-ups to the HOT Z command interpreter.

CSS-P ENter the address of a routine at 5F90, And the PI key causes a jump to that address Enter the address to 5F92, and the TAB key will cause a jump to that address. Addresses entered must not lie in the range 8000-BFFF. See the introduction for an explanation of how to call that memory range.

WRITE Mode Commands

ESCAPE

SS-0 ; key Escapes without change during assembly edit.

HEXEDIT

SS-E Switch to hex-edit mode from assembly edit. Moves the cursor horizontally.

ASSEMBLE

SS-A Switch to assembly-edit mode. Works only when dis-assembly display and edit mode are on. Moves the cursor horizontally. Doesn't work with the data display because assembly doesn't apply to data.

DELETE Deletes the instruction at the cursor and closes up the code between the cursor and END. END may be either lower or higher than the cursor address. If END is less than the cursor address, then code is moved from lower addresses to close the space; if END is greater than the cursor address, then code is moved from higher addresses to close the space. Code at the END address and beyond (moving away from the cursor) is preserved. If END is 256 or more bytes away from the cursor, then you will be asked each time to verify the END value before the command is executed. The purpose of this is to prevent your messing up the entire RAM by forgetting to set END properly.

EDIT Sets the Insert mode for the next instruction (only) to be entered. If END is less than the cursor address, then instructions are pushed to lower addresses (up the screen) as far as END; if END is greater than the cursor address, then instructions are moved to higher addresses (down the screen) as far as END. Any NAMEs assigned to shifted memory area will also be shifted so that they stay with the instruction to which they were assigned. Relative jumps to or from the shifted area are not corrected and may require a fix-up. If END is 256 bytes or more from the cursor address, you will be required to confirm the END value before the operation proceeds.

ENTER Quit to READ mode when cursor is in "home" column. During hex entry, ENTER escapes and leaves the original memory contents intact. During mnemonics entry, ENTER sends the line contents to the assembler for entry into memory.

STEP

STEP Single-steps the instruction at the cursor address and switches to the single-step display with the result of of that instruction in the register values and the following instruction in the NEXT slot.

SET END

TO Brings up the END? cursor that allows you to reset the END variable. Whenever a block of code needs to be marked, it is generally delineated by the cursor address and the address assigned to END. Always use it to block out a segment of memory for Insert and Delete commands before beginning to edit. END should be set within 256 bytes of the cursor for editing, but that restriction can be overridden in any particular case. (See Insert and Delete instructions.)

OR Sets END equal to the current cursor address.

FIND STRING

CSS-F FIND the string marked by the cursor (first byte) and END (last byte). Sets the display to start with the found string. If no match is found, then the display remains at the template string. To find the next match without going back to the template, use CSS-G. Do not use other commands between these two.

FIND NEXT OCCURRENCE OF STRING

CSS-G FINDs the next successive match to the template string set up by CSS-F. After a match is found, you must move the cursor past the beginning of the matching sequence before using this command, to avoid finding the same occurrence again.

ASSIGN NAME

CSS-N NAME command. This command has two separate effects, depending upon whether it is used with the disassembly display or the data display. With the disassembly display, the effect is to christen that instruction with the NAME that you enter to the screen following the command. A NAME requires four characters with at least one beyond F in the alphabet. (All of lower case works.) Space and semicolon should not be used. With the data display, the NAME you enter following the command must already be assigned to some address. HOT Z then looks up the address for that NAME and pokes that address to the byte at the cursor address and the byte following, then moves the cursor down two bytes. Use this form for entering tables of addresses

DELETE NAME

CSS-X Deletes the NAME at the cursor address from the current NAME file. This command will only affect the NAME that you see on screen with the disassembly display, so it is best not to use it with the data display. Do not attempt to use this command before you have moved the NAME file to RAM with the NSET command.

CLEAR MEMORY

ERASE Clears memory from cursor address to END. Works only on RAM.

FILL MEMORY

FN — Fills memory from cursor address to END with the code for a key that you specify in response to the KEY? prompt. For unkeyable values, write that value to the HOT Z variable FILC (5FA4) and then use the ERASE command.

CASSETTE COMMANDS

SAVE CODE

CSS-SAVE — SAVEs code from cursor to END-1. Enter a tape name without quotes. This is a CODE-format SAVE. You can reload such tapes from BASIC by converting the cursor address to decimal and setting the byte length to END minus cursor address. From Home bank only.

VERIFY

VERIFY — VERIFIES a CODE format tape from cursor to END-1. No quotes on tape name. Compares with Home bank.

LOAD CODE

CSS-LOAD — LOAD from cursor to END. Loads 2068 CODE-format tapes. Set the cursor to the start address and END one byte beyond the last, such that END minus cursor address equals the byte length. Unlike the BASIC command and earlier versions of HOT Z, a tape name is always required by this command. No quotes are used. Loads to Home bank.

TRANSFER COMMANDS

CSS-T — Transfers memory content (either within or between banks of memory) between the cursor address and END (inclusive) to a destination (DEST) that you enter following the command. First enter source and destination Banks. (ØØFF means from Dock to Home.) Hit ENTER for a default to FFFF, which means Home-to-Home. Then put in the Destination address (DEST) in the bank you want the stuff to end up in, if that's not too many 'in's.' The ENTER key after DEST executes the command; SPACE after DEST cancels the command; TO after DEST lets you reset END before the command is executed. Does not transfer NAMEs. To do that, use the MERGE command, which is otherwise identical to this one.

CSS-SS-T MERGE) TRANSFER memory contents and assigned NAMEs from a memory block (cursor address to END, inclusive) to an area beginning with an address entered in response to the DEST prompt. (See CSS-T command.) This command depends on the NAME file being in Home RAM; do not attempt to use it until you have done an NSET. (Should NSET be part of initialization?) This command is not so often nexessary, except for small block moves.

DIS/DAT

CSS-GOTO Display switch, data/disassembly. Works only from hex-edit mode. (THEN key) Answers most of your decimal to hex perplexities, reads BASIC and ASCII in rightmost column.

RUN IT

CSS-RUN Runs code beginning at the cursor address. Returns to HOT Z with the first RET. If you do an extra POP and destroy the return address, then you are on your own. (This command expects to jump to the bank structure described by DISB, Home by default, but whatever you set it. If you set a new bank, then then you must set the return which requires a JP back to HOT Z in Bank Ø, chunks 4 and 5.) Recommended procedure is to test your routines first with the single-stepper before attempting the R command.

CHECKSUM

LEN Performs a 32-bit CHECKSUM from cursor address to END and switches to the STEP display, where the sum is in BCDE.

HEX ARITHMETIC

CSS-A Does hex arithmetic. Takes the cursor address (K) and END (E) and displays on the top line the sum (E+K) and difference (E-K) in hexadecimal. Bank indifferent.

PART SCREEN

AT Moves cursor to far left of screen and awaits your entry of an address, then disassembles from that address to bottom of screen. Use it for a composite listing. Use CSS-COPY immediately after to print the screen display. Depends on the Bank-chunk description in DISB for what memory it reads. Therefore, any screen that can be printed will be all in one bank.

CODE RELOCATION COMMANDS

MOVE — Relocates Z80 code between the cursor address and END. Readdresses all CALLs or JPs. Allows a three-way partition of code, variables and (constant) files. Requires nine addresses to be first entered at TEM1 through TEM9. TEM variables are in the permanent NAME file and cohabit with inessential BASIC variables. Set them before you use the command. TEM1 through TEM3 are the start address of the code block, the end address of the code block, and the destination address of the code block. Cursor and End are usually set to the first two of these, and the third is the DEST. TEM4, TEM5, and TEM6 are usually the file block of constants associated with the program, and TEM7, TEM8, and TEM9 are generally the block of variables, or reserved temporary memory space, where the only important thing is the address. HOT Z assumes that these three blocks can be moved independently. If there are blocks you don't want to touch, then you can use 0000 as a default value to any block of three TEM values.

CSS-Y — READDRESS a jump table (address file) between the cursor address and END by a 16-bit displacement value entered in response to the DISP prompt. Takes the address (lo-hi order) at each pair of memory locations, adds the displacement, and re-enters the sum to the same locations.

CSS-U — READDRESS that portion of a NAME file between cursor and END by the value you enter to DISP. For special file manipulations only. Normally, you should use the MERGE command to move NAMEs and code around in memory.

PRINTER COMMANDS

CSS-COPY — COPIES screen to 2040 printer. Intended mainly for use with the PARTSCREEN command for printing out composite disassembly from separate address blocks.

LLIST — Outputs the screen and beyond without headings from the cursor address to END to the 2040 printer.

HOOKUPS

CSS-M
CSS-P — User hook-ups to the HOT Z command interpreter. Enter the address of a routine at 5F98, and the PI key causes a jump to that address. Enter the address to 5F9A, and the TAB key will cause a jump to that address. Addresses entered must NOT lie in the range 8000-BFFF. See the introduction for an explanation of how to call that memory range.

SINGLE-STEP MODE

Key	Function
QUIT	
SS-Q	Quit single-step and return to READ. Return address is the address in the NEXT slot of the single stepper. Register values will be preserved if you reenter from READ mode.
STEP	
ENTER	Runs the instruction in the NEXT slot and reports the resulting register values.
SPACE	Skip the step in the NEXT slot and advance to the next instruction. Skipped instructions are not listed in the LAST slot at the top of the disassembly segment.
EDIT	Backs up. On its first use, this command takes the instruction from the LAST slot at the top of the disassembly listing and puts it in the NEXT slot (second line). Repeated use with no intervening commands will back up one more byte for each keypress. Intended use is just to get the last step back.
PRINTOUT	
CSS-COPY	Print screen. Copies current screen to 2040 printer.
RUN IT	
CSS-RUN	Run a CALL or RST 10. It is your responsibility to know that the called routine will not crash and not to send RST 10 any unprintable characters. The purpose of this command is to shorten the time needed to step through complex routines.
SET BREAKPOINTS	
OR	Set Breakpoint1. Breakpoints are set just as register pairs are, with a NAME or address entry into the NEXT cursor. You must set the breakpoints precisely to the beginning of the instruction at which you want the single-step to stop, because the stop depends on the address of the next step being exactly equal to the breakpoint. If the breakpoint points to the second byte of a two-or-three-byte instruction, you routine will never stop until you crash or hit BREAK.

AT Set Breakpoint2. Breakpoints are set just as register pairs are, with a NAME or address entry into the NEXT cursor. You must set the breakpoints precisely to the beginning of the instruction at which you want the single-step to stop, because the stop depends on the address of the next step being exactly equal to the breakpoint. If the breakpoint points to the second byte of a two-or-three-byte instruction, you routine will never stop until you crash or hit BREAK.

AND Display the two breakpoints on the line below the flags display.

SS-GOTO Go (run) to breakpoint. Causes the test routine to run from the address in the NEXT slot to either of the two breakpoints, which must be set in advance of this command. Breakpoints must be set to an address that starts a command and not to a byte embedded in a command. The GO routine checks the BREAK key after executing each line of code, so you can recover from endless loops and sometimes from runaway routines (if you're quick) by hitting BREAK.

REGISTER SET

VAL Set register value. The response to this command will be REG? in the NEXT cursor. You should respond as follows for the various registers:

A for the A register
B for the BC pair
D for the DE pair
F for the Flags register
H for the HL pair
S for the user's Stack Pointer
X for the IX pointer
Y for the IY pointer

Note that all settings are 16 bits (two bytes) except for the one hex byte for A and the mnemonic setting for F. The specific flag bits are set or reset with the same mnemonics as are reported (M, P, Z, NZ, PO, PE, C, NC). Use this command to set up initial conditions for testing your routines. Note that you can set the user's SP this way.

ASSEMBLY

SS-A Sets the assembly cursor at the instruction in the NEXT slot so that you can EDIT it. Return to STEP operation with ENTER.

SPECIAL DISPLAY SCREEN

ATTR — SETs a second display file (WINDOW) starting at the address in NEXT and extending 1B00 bytes. Any stepped display instructions then output to the window, which comes up before the next register display. Be careful not to erase valuable code by setting the window on top of it. Dismiss the screen with any key but V.

SCR$ — Toggles the feature that causes the WINDOW to wait for a keystroke before going to register display.

OUT — Switches the window out of the STEP loop so that subsequent steps have no effect on it.

IN — Switches a WINDOW from OUT back IN again. WINDOW must be SET up first.

HOOKUPS

CSS-M
CSS-P — User hook-ups to the hot z command interpreter. Enter the address of a routine at 5F94, and the PI key cause a jump to that address. Enter the address to 5F96, and the TAB key will cause a jump to that address. Addresses entered must not lie in the range 8000-BFFF. See the introduction for an explanation of how to call that memory range.